(Sub)Gradient Descent

CMSC 422
MARINE CARPUAT
marine@cs.umd.edu

Figures credit: Piyush Rai
Logistics

- Midterm is on Thursday 3/24
  - during class time
  - closed book/internet/etc, one page of notes.
  - will include short questions (similar to quizzes) and 2 problems that require applying what you've learned to new settings
  - topics: everything up to this week, including linear models, gradient descent, homeworks and project 1

- Next HW due on Tuesday 3/22 by 1:30pm
- Office hours Tuesday 3/22 after class
- Please take survey before end of break!
What you should know (1)

Decision Trees
• What is a decision tree, and how to induce it from data

Fundamental Machine Learning Concepts
• Difference between memorization and generalization
• What inductive bias is, and what is its role in learning
• What underfitting and overfitting means
• How to take a task and cast it as a learning problem

• Why you should never ever touch your test data!!
What you should know (2)

• New Algorithms
  – K-NN classification
  – K-means clustering

• Fundamental ML concepts
  – How to draw decision boundaries
  – What decision boundaries tells us about the underlying classifiers
  – The difference between supervised and unsupervised learning
What you should know (3)

• The perceptron model/algorithm
  – What is it? How is it trained? Pros and cons? What guarantees does it offer?
  – Why we need to improve it using voting or averaging, and the pros and cons of each solution

• Fundamental Machine Learning Concepts
  – Difference between online vs. batch learning
  – What is error-driven learning
What you should know (4)

• Be aware of practical issues when applying ML techniques to new problems

• How to select an appropriate evaluation metric for imbalanced learning problems

• How to learn from imbalanced data using $\alpha$-weighted binary classification, and what the error guarantees are
What you should know (5)

• What are reductions and why they are useful

• Implement, analyze and prove error bounds of algorithms for
  – Weighted binary classification
  – Multiclass classification (OVA, AVA, tree)

• Understand algorithms for
  – Stacking for collective classification
  – $\omega$ – ranking
What you should know (6)

• Linear models:
  – An optimization view of machine learning
  – Pros and cons of various loss functions
  – Pros and cons of various regularizers

• (Gradient Descent)
Today’s topic

How to optimize linear model objectives using gradient descent (and subgradient descent)

[CIML Chapter 6]
Casting Linear Classification as an Optimization Problem

Objective function

Loss function
measures how well classifier fits training data

Regularizer
prefers solutions that generalize well

\[
\min_{w, b} L(w, b) = \min_{w, b} \sum_{n=1}^{N} \mathbb{I}(y_n(w^T x_n + b) < 0) + \lambda R(w, b)
\]

\(\mathbb{I}(\cdot)\) Indicator function: 1 if (\(\cdot\)) is true, 0 otherwise

The loss function above is called the 0-1 loss
Gradient descent

• A general solution for our optimization problem

\[
\min_{w,b} L(w,b) = \min_{w,b} \sum_{n=1}^{N} \mathbb{I}(y_n(w^T x_n + b) < 0) + \lambda R(w, b)
\]

• Idea: take iterative steps to update parameters in the direction of the gradient
Gradient descent algorithm

Algorithm 2.2 \texttt{GradientDescent}(F, K, \eta_1, \ldots)

1: \( z^{(0)} \leftarrow \langle 0, 0, \ldots, 0 \rangle \)  
   \hspace{1cm} // initialize variable we are optimizing
2: \textbf{for} \( k = 1 \ldots K \) \textbf{do}  
3: \hspace{1cm} \( g^{(k)} \leftarrow \nabla_z F|_{z^{(k-1)}} \)  
   \hspace{1cm} // compute gradient at current location
4: \hspace{1cm} \( z^{(k)} \leftarrow z^{(k-1)} - \eta^{(k)} g^{(k)} \)  
   \hspace{1cm} // take a step down the gradient
5: \textbf{end for}  
6: \textbf{return} \( z^{(K)} \)
Illustrating gradient descent in 1-dimensional case
Gradient Descent

• 2 questions

  – When to stop?

  – How to choose the step size?
Gradient Descent

• 2 questions
  – When to stop?
    • When the gradient gets close to zero
    • When the objective stops changing much
    • When the parameters stop changing much
    • Early
    • When performance on held-out dev set plateaus
  – How to choose the step size?
    • Start with large steps, then take smaller steps
Now let’s calculate gradients for multivariate objectives

• Consider the following learning objective

\[ \mathcal{L}(w, b) = \sum_n \exp \left[ - y_n (w \cdot x_n + b) \right] + \frac{\lambda}{2} \|w\|^2 \]

• What do we need to do to run gradient descent?
(1) Derivative with respect to \( b \)

\[
\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial}{\partial b} \sum_n \exp \left[ - y_n (\mathbf{w} \cdot \mathbf{x}_n + b) \right] + \frac{\partial}{\partial b} \frac{\lambda}{2} \| \mathbf{w} \|^2
\]

\[
= \sum_n \frac{\partial}{\partial b} \exp \left[ - y_n (\mathbf{w} \cdot \mathbf{x}_n + b) \right] + 0
\]

\[
= \sum_n \left( \frac{\partial}{\partial b} - y_n (\mathbf{w} \cdot \mathbf{x}_n + b) \right) \exp \left[ - y_n (\mathbf{w} \cdot \mathbf{x}_n + b) \right]
\]

\[
= - \sum_n y_n \exp \left[ - y_n (\mathbf{w} \cdot \mathbf{x}_n + b) \right]
\]
(2) Gradient with respect to \( w \)

\[
\nabla_w \mathcal{L} = \nabla_w \sum_n \exp \left[ -y_n (w \cdot x_n + b) \right] + \nabla_w \frac{\lambda}{2} ||w||^2
\]

\[
= \sum_n (\nabla_w - y_n (w \cdot x_n + b)) \exp \left[ -y_n (w \cdot x_n + b) \right] + \lambda w
\]

\[
= -\sum_n y_n x_n \exp \left[ -y_n (w \cdot x_n + b) \right] + \lambda w
\]
Subgradients

• Problem: some objective functions are not differentiable everywhere
  – Hinge loss, l1 norm

• Solution: subgradient optimization
  – Let’s ignore the problem, and just try to apply gradient descent anyway!!
  – we will just differentiate by parts...
Example: subgradient of hinge loss

$$\partial_w \max \{0, 1 - y_n (w \cdot x_n + b)\}$$  \hspace{1cm} (6.22)

$$= \partial_w \begin{cases} 
0 & \text{if } y_n (w \cdot x_n + b) > 1 \\
y_n (w \cdot x_n + b) & \text{otherwise}
\end{cases}$$  \hspace{1cm} (6.23)

$$= \begin{cases} 
\partial_w 0 & \text{if } y_n (w \cdot x_n + b) > 1 \\
\partial_w y_n (w \cdot x_n + b) & \text{otherwise}
\end{cases}$$  \hspace{1cm} (6.24)

$$= \begin{cases} 
0 & \text{if } y_n (w \cdot x_n + b) > 1 \\
y_n x_n & \text{otherwise}
\end{cases}$$  \hspace{1cm} (6.25)
Subgradient Descent for Hinge Loss

**Algorithm 23**  

**HingeRegularizedGD(D, \( \lambda \), MaxIter)**

1. \( w \leftarrow \langle 0,0,\ldots,0 \rangle \), \( b \leftarrow 0 \)  
   // initialize weights and bias

2. **for** iter = 1 \ldots MaxIter **do**

3. \( g \leftarrow \langle 0,0,\ldots,0 \rangle \), \( g \leftarrow 0 \)  
   // initialize gradient of weights and bias

4. **for all** \((x,y) \in D\) **do**

5. \( y(w \cdot x + b) \leq 1 \) **then**

6. \( g \leftarrow g + yx \)  
   // update weight gradient

7. \( g \leftarrow g + y \)  
   // update bias derivative

8. **end if**

9. **end for**

10. \( g \leftarrow g - \lambda w \)  
    // add in regularization term

11. \( w \leftarrow w + \eta g \)  
    // update weights

12. \( b \leftarrow b + \eta g \)  
    // update bias

13. **end for**

14. **return** \( w, b \)
Summary

• Gradient descent
  – A generic algorithm to minimize objective functions
  – Works well as long as functions are well behaved (i.e. convex)
  – Subgradient descent can be used at points where derivative is not defined
  – Choice of step size is important

• Optional: can we do better?
  – For some objectives, we can find closed form solutions (see CIML 6.6)